1 Static Semantics

$\boxed{\Theta; \Delta; \Gamma \vdash e : t}$ Typing rules for expressions

$$\frac{}{\Theta; \Delta; \cdot, x : t \vdash x : t} \quad \textsc{Ty\_Var\_Lin}$$

$$\frac{x : t \in \Delta}{\Theta; \Delta; \cdot \vdash x : t} \quad \textsc{Ty\_Var}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma', x : t \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, x = e\, \mathbf{in}\, e' : t'} \quad \textsc{Ty\_Let}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash () : \mathbf{unit}} \quad \textsc{Ty\_Unit\_Intro}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : \mathbf{unit} \\ \Theta; \Delta; \Gamma' \vdash e' : t\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, () = e\, \mathbf{in}\, e' : t} \quad \textsc{Ty\_Unit\_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{true} : \mathbf{bool}} \quad \textsc{Ty\_Bool\_True}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{false} : \mathbf{bool}} \quad \textsc{Ty\_Bool\_False}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : !\mathbf{bool} \\ \Theta; \Delta; \Gamma' \vdash e_1 : t' \\ \Theta; \Delta; \Gamma' \vdash e_2 : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{if}\, e\, \mathbf{then}\, e_1\, \mathbf{else}\, e_2 : t} \quad \textsc{Ty\_Bool\_Elim}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash k : \mathbf{int}} \quad \textsc{Ty\_Int\_Intro}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash el : \mathbf{elt}} \quad \textsc{Ty\_Elt\_Intro}$$

$$\frac{\begin{array}{l}\Theta; \Delta; \cdot \vdash v : t \\ v \neq l\end{array}}{\Theta; \Delta; \cdot \vdash \mathbf{Many}\, v : !t} \quad \textsc{Ty\_Bang\_Intro}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : !t \\ \Theta; \Delta, x : t; \Gamma' \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, \mathbf{Many}\, x = e\, \mathbf{in}\, e' : t'} \quad \textsc{Ty\_Bang\_Elim}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma' \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash (e, e') : t \otimes t'} \quad \textsc{Ty\_Pair\_Intro}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e_{12} : t_1 \otimes t_2 \\ \Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e : t\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let}\, (a, b) = e_{12}\, \mathbf{in}\, e : t} \quad \textsc{Ty\_Pair\_Elim}$$

$$\frac{\begin{array}{c}\Theta \vdash t'\ \mathsf{Type} \\ \Theta; \Delta; \Gamma, x : t' \vdash e : t\end{array}}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, x : t' \to e : t' \multimap t} \quad \textsc{Ty\_Lambda}$$

$$\frac{\begin{array}{c}\Theta; \Delta; \Gamma \vdash e : t' \multimap t \\ \Theta; \Delta; \Gamma' \vdash e' : t'\end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash e\, e' : t} \quad \textsc{Ty\_App}$$

$$\frac{\Theta, fc; \Delta; \Gamma \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun}\, fc \to e : \forall fc.t} \quad \textsc{Ty\_Gen}$$

$$\frac{\begin{array}{l}\Theta \vdash f\ \mathsf{Cap} \\ \Theta; \Delta; \Gamma \vdash e : \forall fc.t\end{array}}{\Theta; \Delta; \Gamma \vdash e[f] : t[f/fc]} \quad \textsc{Ty\_Spc}$$

$$\frac{\Theta; \Delta, g : t \multimap t'; \cdot, x : t \vdash e : t'}{\Theta; \Delta; \cdot \vdash \mathbf{fix}\,(g, x : t, e : t') : t \multimap t'} \quad \textsc{Ty\_Fix}$$

2 Dynamic Semantics

$\boxed{\langle \sigma, e \rangle \to \mathit{StepsTo}}$ operational semantics

$$\frac{}{\langle \sigma, \mathbf{let}\,() = ()\,\mathbf{in}\,e \rangle \to \langle \sigma, e \rangle} \quad \textsc{Op\_Let\_Unit}$$

$$\frac{}{\langle \sigma, \mathbf{let}\,x = v\,\mathbf{in}\,e \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_Let\_Var}$$

$$\frac{}{\langle \sigma, \mathbf{if}\,(\mathbf{Many\,true})\,\mathbf{then}\,e_1\,\mathbf{else}\,e_2 \rangle \to \langle \sigma, e_1 \rangle} \quad \textsc{Op\_If\_True}$$

$$\frac{}{\langle \sigma, \mathbf{if}\,(\mathbf{Many\,false})\,\mathbf{then}\,e_1\,\mathbf{else}\,e_2 \rangle \to \langle \sigma, e_2 \rangle} \quad \textsc{Op\_If\_False}$$

$$\frac{}{\langle \sigma, \mathbf{let\,Many}\,x = \mathbf{Many}\,v\,\mathbf{in}\,e \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_Let\_Many}$$

$$\frac{}{\langle \sigma, \mathbf{let}\,(a, b) = (v_1, v_2)\,\mathbf{in}\,e \rangle \to \langle \sigma, e[a/v_1][b/v_2] \rangle} \quad \textsc{Op\_Let\_Pair}$$

$$\frac{}{\langle \sigma, (\mathbf{fun}\,fc \to v)[f] \rangle \to \langle \sigma, v[fc/f] \rangle} \quad \textsc{Op\_Frac\_Cap}$$

$$\frac{}{\langle \sigma, \mathbf{fix}\,(g, x : t, e : t')\,v \rangle \to \langle \sigma, e[x/v][g/\mathbf{fix}\,(g, x : t, e : t')] \rangle} \quad \textsc{Op\_App\_Fix}$$

$$\frac{}{\langle \sigma, (\mathbf{fun}\,x : t \to e)\,v \rangle \to \langle \sigma, e[x/v] \rangle} \quad \textsc{Op\_App\_Lambda}$$

$$\frac{\langle \sigma, e \rangle \to \langle \sigma', e' \rangle}{\langle \sigma, C[e] \rangle \to \langle \sigma, C[e'] \rangle} \quad \textsc{Op\_Context}$$

$$\frac{\langle \sigma, e \rangle \to \mathbf{err}}{\langle \sigma, C[e] \rangle \to \mathbf{err}} \quad \textsc{Op\_Context\_Err}$$

$$\frac{0 \le k_1, k_2}{\langle \sigma, \mathbf{matrix}\,k_1\,k_2 \rangle \to \langle \sigma \uplus \{l \mapsto_1 M_{k_1,k_2}\}, l \rangle} \quad \textsc{Op\_Matrix}$$

$$\frac{}{\langle \sigma \uplus \{l \mapsto_1 m_{k_1,k_2}\}, \mathbf{free}\,l \rangle \to \langle \sigma, () \rangle} \quad \textsc{Op\_Free}$$

$$\frac{}{\langle \sigma \uplus \{l \mapsto_f m_{k_1,k_2}\}, \mathbf{share}\,l \rangle \to \langle \sigma \uplus \{l \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\} \uplus \{l \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\}, (l, l) \rangle} \quad \textsc{Op\_Share}$$

$$\frac{f \le 1}{\langle \sigma \uplus \{l \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\} \uplus \{l \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\}, \mathbf{unshare}\,l\,l \rangle \to \langle \sigma \uplus \{l \mapsto_f m_{k_1,k_2}\}, l \rangle} \quad \textsc{Op\_Unshare\_Eq}$$

$$\frac{l \ne l'}{\langle \sigma \uplus \{l \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\} \uplus \{l' \mapsto_{\frac{1}{2}\cdot f} m_{k_1,k_2}\}, \mathbf{unshare}\,l\,l' \rangle \to \mathbf{err}} \quad \textsc{Op\_Unshare\_Neq}$$

$$\frac{\sigma' = \sigma \uplus \{l_1 \mapsto_{fc_1} m_{1\,k_1,k_2}\} \uplus \{l_2 \mapsto_{fc_2} m_{2\,k_2,k_3}\}}{\langle \sigma' \uplus \{l_3 \mapsto_1 m_{1\,k_1,k_3}\}, \mathbf{gemm}\,l_1\,l_2\,l_3 \rangle \to \langle \sigma' \uplus \{l_3 \mapsto_1 (m_1\,m_2 + m_3)_{k_1,k_3}\}, ((l_1, l_2), l_3) \rangle} \quad \textsc{Op\_Gemm\_Match}$$

$$\frac{\begin{array}{l} k_2 \ne k_2' \\ \sigma' = \sigma \uplus \{l_1 \mapsto_{fc_1} m_{1\,k_1,k_2}\} \uplus \{l_2 \mapsto_{fc_2} m_{2\,k_2',k_3}\} \end{array}}{\langle \sigma' \uplus \{l_3 \mapsto_1 m_{1\,k_1,k_3}\}, \mathbf{gemm}\,l_1\,l_2\,l_3 \rangle \to \mathbf{err}} \quad \textsc{Op\_Gemm\_Mismatch}$$

3 Interpretation

3.1 Definitions

Let $Heap \equiv Loc \rightharpoonup Permission \times Matrix$ and $\star : Heap \times Heap \rightharpoonup Heap$ as follows:

$$(\sigma_1 \star \sigma_2)(l) \equiv \begin{cases} \sigma_1(l) & \text{if } l \in \text{dom}(\sigma_1) \wedge l \notin \text{dom}(\sigma_2) \\ \sigma_2(l) & \text{if } l \in \text{dom}(\sigma_2) \wedge l \notin \text{dom}(\sigma_1) \\ (f_1 + f_2, m) & \text{if } (f_1, m) = \sigma_1(l) \wedge (f_2, m) = \sigma_2(l) \wedge f_1 + f_2 \leq 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Commutativity and associativity of $\star$ follows from that of $+$.
$\sigma_1 \star \sigma_2$ is "defined" if it is for all $l \in \text{dom}(\sigma_1) \cup \text{dom}(\sigma_2)$.

The $n-$fold iteration for the (functional) *StepsTo* relation, is also a (functional) relation:

$$\begin{aligned} \forall n.\ \mathbf{err} &\to^n \mathbf{err} \\ \forall n.\ \langle \sigma, v \rangle &\to^n \langle \sigma, v \rangle \\ \langle \sigma, e \rangle &\to^0 \langle \sigma, e \rangle \\ \langle \sigma, e \rangle &\to^{n+1} ((\langle \sigma, e \rangle \to) \to^n) \end{aligned}$$

Hence, all bounded iterations end in either an **err**, a heap-and-expression or a heap-and-value.

3.2 Interpretation

$$\mathcal{V}_k[\![\mathbf{unit}]\!] = \{(\emptyset, *)\}$$

$$\mathcal{V}_k[\![\mathbf{bool}]\!] = \{(\emptyset, true), (\emptyset, false)\}$$

$$\mathcal{V}_k[\![\mathbf{int}]\!] = \{(\emptyset, n) \mid 2^{-63} \leq n \leq 2^{63} - 1\}$$

$$\mathcal{V}_k[\![\mathbf{elt}]\!] = \{(\emptyset, f) \mid f \text{ a IEEE Float64 }\}$$

$$\mathcal{V}_k[\![f\,\mathbf{mat}]\!] = \{(\{l \mapsto_{2^{-f}} \_\}, l)\}$$

$$\mathcal{V}_k[\![!t]\!] = \{(\emptyset, \mathbf{Many}\, v) \mid (\emptyset, v) \in \mathcal{V}_k[\![t]\!]\}$$

$$\mathcal{V}_k[\![\forall fc.\ t]\!] = \{(\sigma, \mathbf{fun}\, fc \to\ v) \mid \forall f.\ (\sigma, (\mathbf{fun}\, fc \to\ v)\,[f]) \in \mathcal{V}_k[\![t[fc/f]]\!]\}$$

$$\mathcal{V}_k[\![t_1 \otimes t_2]\!] = \{(\sigma_1 \star \sigma_2, (v_1, v_2)) \mid (\sigma_1, v_1) \in \mathcal{V}_k[\![t_1]\!] \wedge (\sigma_2, v_2) \in \mathcal{V}_k[\![t_2]\!]\}$$

$$\begin{aligned}\mathcal{V}_k[\![t \multimap t']\!] = \{(\sigma, v') \mid\ & (v' = \mathbf{fun}\, x : t \to e \vee v' = \mathbf{fix}(g, x : t, e : t')) \wedge \\ & \forall j < k, (\sigma_v, v) \in \mathcal{V}_j[\![t]\!].\ \sigma \star \sigma_v \text{ defined } \Rightarrow (\sigma \star \sigma_v, v'\,v) \in \mathcal{C}_j[\![t']\!]\}\end{aligned}$$

$$\begin{aligned}\mathcal{C}_k[\![t]\!] = \{(\sigma_s, e) \mid\ & \forall\, j \leq k, \sigma_r.\ \sigma_s \star \sigma_r \text{ defined } \Rightarrow \langle \sigma_s \star \sigma_r, e\rangle \to^j \mathbf{err}\ \vee \exists \sigma_f, e'. \\ & \langle \sigma_s \star \sigma_r, e\rangle \to^j \langle \sigma_f \star \sigma_r, e'\rangle \wedge (e' \text{ is a value } \Rightarrow (\sigma_f \star \sigma_r, e') \in \mathcal{V}_{k-j}[\![t]\!])\}\end{aligned}$$

$$\mathcal{I}_k[\![\cdot]\!]\theta = \{[]\}$$

$$\mathcal{I}_k[\![\Delta, x : t]\!]\theta = \{\delta[x \mapsto v_x] \mid \delta \in \mathcal{I}_k[\![\Delta]\!]\theta \wedge (\emptyset, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]\}$$

$$\mathcal{L}_k[\![\cdot]\!]\theta = \{(\emptyset, [])\}$$

$$\mathcal{L}_k[\![\Gamma, x : t]\!]\theta = \{(\sigma \star \sigma_x, \gamma[x \mapsto v_x]) \mid (\sigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta \wedge (\sigma_x, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]\}$$

$$\begin{aligned}{}_k[\![\Theta; \Delta; \Gamma \vdash e : t]\!] = \forall \theta, \delta, \gamma, \sigma.\ & \mathrm{dom}(\Theta) = \mathrm{dom}(\theta) \wedge (\sigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta \wedge \delta \in \mathcal{I}_k[\![\Delta]\!]\theta \Rightarrow \\ & (\sigma, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]\end{aligned}$$

4 Proofs

4.1 Lemmas

$$\mathcal{V}_k[\![\theta(t)]\!] \subseteq \mathcal{C}_k[\![\theta(t)]\!]$$

$$\langle \sigma, e \rangle \to^* = \langle \sigma \star \sigma_r, e \rangle \to^*$$

$$\delta(\gamma(v)) \text{ is a value.}$$

$$j \leq k \Rightarrow \_\,_k[\![\cdot]\!] \subseteq \_\,_j[\![\cdot]\!]$$

$$(\sigma', e') \in \mathcal{C}_{k-1}[\![\cdot]\!] \wedge \langle \sigma, e \rangle \to \langle \sigma', e' \rangle \Rightarrow (\sigma, e) \in \mathcal{C}_k[\![\cdot]\!]$$

4.2 Soundness

$$\forall \Theta, \Delta, \Gamma, e, t.\ \Theta; \Delta; \Gamma \vdash e : t \Rightarrow \forall k.\ _k[\![\Theta; \Delta; \Gamma \vdash e : t]\!]$$

PROOF SKETCH: Induction over the typing judgements.

ASSUME: 1. Arbitrary $\Theta, \Delta, \Gamma, e, t$ such that $\Theta; \Delta; \Gamma \vdash e : t$.
2. Arbitrary $\theta, k, \delta, \gamma, \sigma$ such that:
a. $\text{dom}(\Theta) = \text{dom}(\theta)$
b. $(\sigma, \gamma) \in \mathcal{L}_k[\![\Gamma]\!]\theta$
c. $\delta \in \mathcal{I}_k[\![\Delta]\!]\theta$.
3. W.l.o.g., all variables are distinct/dom(Δ) and dom(Γ) are disjoint.
4. And so that over expressions $\gamma \circ \delta = \delta \circ \gamma$.
5. By construction, $\text{dom}(\Delta) = \text{dom}(\delta)$ and $\text{dom}(\Gamma) = \text{dom}(\gamma)$.

PROVE: $(\sigma, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t')]\!]$.
ASSUME: Arbitrary $j \leq k$ and σ_r.
SUFFICES: Show whole expression either reduces to **err** or takes j steps.

$\langle 1 \rangle$1. CASE: TY_LET.
PROVE: $(\sigma, \gamma(\delta(\mathbf{let}\, x \;=\; e\, \mathbf{in}\, e'))) \in \mathcal{C}_k[\![\theta(t')]\!]$.
SUFFICES: $(\sigma, \mathbf{let}\, x \;=\; \gamma(\delta(e))\, \mathbf{in}\, \gamma(\delta(e'))) \in \mathcal{C}_k[\![\theta(t')]\!]$.

$\langle 2 \rangle$1. By induction,
1. $[\![\Theta; \Delta; \Gamma \vdash e : t]\!]$
2. $[\![\Theta; \Delta; \Gamma', x : t \vdash e' : t']\!]$.

$\langle 2 \rangle$2. By 2b and induction on Γ', we know there exist $\sigma_{e'}$, $(\sigma_e, \gamma_e) \in \mathcal{L}_k[\![\Gamma]\!]$, such that $\sigma = \sigma_e \star \sigma_{e'}$.

$\langle 2 \rangle$3. So, using them, θ, k, δ, and 3 we have $(\sigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

$\langle 2 \rangle$4. By 3, $(\sigma_e, \gamma(\delta(e))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

⟨2⟩5. By definition of $\mathcal{C}_k[\![\cdot]\!]$ and ⟨2⟩2, we instantiate with j and $\sigma_r = \sigma_{e'}$ to conclude that $\langle \sigma, \gamma(\delta(e)) \rangle$ either reduces to **err** or another heap and expression.

⟨2⟩6. Case: **err**
??? By Op_Context_Err and 3, the whole expression reduces to **err** in $j \leq k$ steps. Since $j \leq k$ and σ_r (for 4.1) are arbitrary, $(\sigma, \gamma(\delta(\mathbf{let}\, x \;=\; e\, \mathbf{in}\, e'))) \in \mathcal{C}_k[\![\theta(t')]\!]$.

⟨2⟩7. Case: j steps to another heap and expression.
By Op_Context and 3, the whole expression does the same.

⟨2⟩8. If it is not a value, we are done. ??? If it is $(\sigma_{ef}, v) \in \mathcal{V}_{k-j}[\![\theta(t)]\!]$ by 4.1.
Suffices: $(\sigma_{ef} \star \sigma_{e'}, \mathbf{let}\, x \;=\; v\, \mathbf{in}\, \gamma(\delta(e'))) \in \mathcal{C}_{k-j}[\![\theta(t')]\!]$.
Suffices: ??? $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.

⟨2⟩9. Define: $\gamma_{e'}(y) = v$ if $y = x$ and $\gamma(y)$ if $y \in \mathrm{dom}(\Gamma')$.
??? Thus, by 4.1, $(\sigma_{e'}, \gamma_{e'}) \in \mathcal{L}_k[\![\Gamma', x : t]\!]\theta \subseteq \mathcal{L}_{k-j-1}[\![\Gamma', x : t]\!]\theta$.

⟨2⟩10. Instantiate 2 of step ⟨2⟩1 with $\theta, k-j-1, \delta, \gamma_{e'}, \sigma_{e'}$ to conclude $(\sigma_{e'}, \gamma_{e'}(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨2⟩11. By 3, we have $\gamma(\delta(e'))[x/v] = \gamma_{e'}(\delta(e'))$ and by 4.1 we conclude $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$

⟨1⟩2. Case: Ty_Pair_Elim.
Prove: $(\sigma, \gamma(\delta(\mathbf{let}\,(a, b) \;=\; e\, \mathbf{in}\, e'))) \in \mathcal{C}_k[\![\theta(t')]\!]$.
Proof: Similar to Ty_Let but with Op_Let_Pair

⟨2⟩1. When $(\sigma_{ef}, v) \in \mathcal{V}_{k-j}[\![\theta(t_1) \otimes \theta(t_2)]\!]$, we have $v = (v_1, v_2)$.

⟨2⟩2. Suffices: ??? $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.

⟨2⟩3. Define: $\gamma_{e'}$ to be the restriction of γ to $\mathrm{dom}(\Gamma')$.
??? Thus, by 4.1, $(\sigma_{e'}, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2]) \in \mathcal{L}_k[\![\Gamma', a : t_1, b : t_2]\!]\theta$
$\subseteq \mathcal{L}_{k-j-1}[\![\Gamma', a : t_1, b : t_2]\!]\theta$.

⟨2⟩4. Instantiate $[\![\Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e' : t']\!]$ with $\theta, k-j-1, \delta, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2], \sigma_{e'}$.

⟨2⟩5. ??? By 3 $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨1⟩3. Case: Ty_Bang_Elim.
Prove: $(\sigma, \gamma(\delta(\mathbf{let\, Many}\, x \;=\; e\, \mathbf{in}\, e'))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
Proof sketch: Similar to Ty_Let, but with the following key differences.

⟨2⟩1. When $(\sigma_{ef}, v) \in \mathcal{V}_{k-j}[\![\theta(!t)]\!]$, since $\mathcal{V}_{k-j}[\![\theta(!t)]\!] = \mathcal{V}_{k-j}[\![!\theta(t)]\!]$,
we have $\sigma_{ef} = \emptyset$ and $v = \mathbf{Many}\, v'$ for some $(\emptyset, v') \in \mathcal{V}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩2. Suffices: $(\sigma_{e'}, \mathbf{let\, Many}\, x \;=\; \mathbf{Many}\, v'\, \mathbf{in}\, \gamma(\delta(e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩3. Suffices: $(\sigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t)]\!]$.

⟨2⟩4. Define: $\gamma_{e'}$ as the restriction of γ to $\mathrm{dom}(\Gamma')$.

⟨2⟩5. Instantiate $[\![\Theta; \Delta, x : t, \Gamma' \vdash e' : t']\!]$ with $\theta, k-j-1, \delta_{e'} = \delta[x \mapsto v'], \gamma_{e'}, \sigma_{e'}$ to conclude $(\sigma_{e'}, \gamma_{e'}(\delta_{e'}(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t)]\!]$.

⟨2⟩6. ??? By 3, $(\sigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-j-1}[\![\theta(t)]\!]$.

⟨1⟩4. CASE: TY_UNIT_ELIM.
PROVE: $(\sigma, \gamma(\delta(\textbf{let}\,() \;=\; e\,\textbf{in}\,e'))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
PROOF: Similar to TY_LET but with OP_LET_UNIT.

⟨2⟩1. When $(\sigma_{ef}, v) \in \mathcal{V}_{k-j}[\![\textbf{unit}]\!]$, we have $\sigma_{ef} = \emptyset$ and $v = ()$.

⟨2⟩2. SUFFICES: ??? $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$ by 4.1.

⟨2⟩3. DEFINE: $\gamma_{e'}$ to be the restriction of γ to $\mathrm{dom}(\Gamma')$.
??? Thus, by 4.1, $(\sigma_{e'}, \gamma_{e'}) \in \mathcal{L}_k[\![\Gamma']\!]\theta \subseteq \mathcal{L}_{k-j-1}[\![\Gamma']\!]\theta$.

⟨2⟩4. Instantiate $[\![\Theta; \Delta; \Gamma' \vdash e' : t']\!]$ with $\theta, k-j-1, \delta, \gamma_{e'}, \sigma_{e'}$.

⟨2⟩5. ??? By 3 $(\sigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-j-1}[\![\theta(t')]\!]$.

⟨1⟩5. CASE: TY_BOOL_ELIM.
PROVE: $(\sigma, \gamma(\delta(\textbf{if}\,e\,\textbf{then}\,e_1\,\textbf{else}\,e_2))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
PROOF: Similar to TY_UNIT_ELIM but with OP_IF_{TRUE,FALSE}
and $\sigma_{ef} = \emptyset$ and $v = \textbf{Many true}$ or $v = \textbf{Many false}$.

⟨1⟩6. CASE: TY_BANG_INTRO.
PROVE: $(\sigma, \gamma(\delta(\textbf{Many}\,e))) \in \mathcal{C}_k[\![\theta(!t)]\!]$.
SUFFICES: $(\sigma, \textbf{Many}\,\gamma(\delta(e))) \in \mathcal{C}_k[\![!\theta(t)]\!]$.

⟨2⟩1. By assumption of TY_BANG_INTRO, $e = v$ for some value $v \neq l$, $\Gamma = \emptyset$ and so
$[\![\Theta; \Delta; \cdot \vdash v : t]\!]$ by induction.

⟨2⟩2. SUFFICES: $(\emptyset, \textbf{Many}\,\delta(v)) \in \mathcal{C}_k[\![!\theta(t)]\!]$ by 3 and 2b.

⟨2⟩3. Instantiate $[\![\Theta; \Delta; \cdot \vdash v : t]\!]$ with $\theta, k, \delta, \gamma = [], \sigma = \emptyset$ to obtain $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$.

⟨2⟩4. Instantiate $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$ with $j = 0$, and $\sigma_r = \emptyset$, to conclude $(\emptyset, v) \in \mathcal{V}_k[\![\theta(t)]\!]$.

⟨2⟩5. ??? By definition of $\mathcal{V}_k[\![!\theta(t)]\!]$, 4.1 and 4.1 we have $(\emptyset, \textbf{Many}\,\delta(v)) \in \mathcal{C}_k[\![!\theta(t)]\!]$.

⟨1⟩7. CASE: TY_PAIR_INTRO.
PROVE: $(\sigma, \gamma(\delta(\,(e, e')\,))) \in \mathcal{C}_k[\![\theta(t \otimes t')]\!]$.
ASSUME: Arbitrary $j \leq k$ and σ_r.
SUFFICES: Show whole expression either reduces to **err** or a heap and expression in j steps.

⟨2⟩1. DEFINE: $(\sigma_1, \gamma_1) \in \mathcal{L}_j[\![\Gamma]\!]$ similar to (σ_e, γ_e) in TY_LET.

⟨2⟩2. By induction,
1. $[\![\Theta; \Delta; \Gamma_1 \vdash e_1 : t_1]\!]$
2. $[\![\Theta; \Delta; \Gamma_2 \vdash e_2 : t_2]\!]$.

⟨2⟩3. Instantiate the first with $\theta, k, \delta, \gamma_1, \sigma_1$.

⟨2⟩4. Therefore, $(\sigma_1, \gamma_1(\delta(e_1))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

⟨2⟩5. So, $(\sigma_1 \star \sigma_2, \gamma_1(\delta(e_1)))$ either reduces to **err** or a heap and expression in j steps.

⟨2⟩6. CASE: **err**
??? By OP_CONTEXT_ERR and 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1) are arbitrary, $(\sigma, \gamma(\delta(\,(e, e')\,))) \in \mathcal{C}_k[\![\theta(t \otimes t')]\!]$.

⟨2⟩7. Case: j steps to another heap and expression.
By Op_Context and 3, the whole expression does the same.

⟨2⟩8. If it is not a value, we are done. ??? If it is $(\sigma_{1f}, v_1) \in \mathcal{V}_{k-j}[\![\theta(t_1)]\!]$ by 4.1.
Suffices: ??? By 4.1, $(\sigma_{1f} \star \sigma_{e_2},\ (v_1, e_2)\,) \in \mathcal{C}_{k-j}[\![\theta(t_1 \otimes t_2)]\!]$.

⟨2⟩9. Instantiate the second IH with $\theta, j, \delta, \gamma_2, \sigma_2$ defined as per usual.

⟨2⟩10. So, $(\sigma_{1f} \star \sigma_2, \gamma_2(\delta(e_2)))$ either reduces to **err** or a heap and expression in j steps.

⟨2⟩11. Case: **err**
??? By Op_Context_Err, 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1) are arbitrary, $(\sigma_{e_2},\ (v_1, e_2)\,) \in \mathcal{C}_{k-j}[\![\theta(t_1 \otimes t_2)]\!]$.

⟨2⟩12. Case: j steps to another heap and expression.
By Op_Context and 3, the whole expression does the same.

⟨2⟩13. If it is not a value, we are done. ??? If it is $(\sigma_{2f}, v_2) \in \mathcal{V}_{k-j}[\![\theta(t_2)]\!]$ by 4.1.
Suffices: ??? By 4.1, $(\sigma_{1f} \star \sigma_{2f},\ (v_1, v_2)\,) \in \mathcal{C}_{k-2j}[\![\theta(t_1 \otimes t_2)]\!]$.

⟨2⟩14. ??? By 4.1 and 4.1, $(\sigma_{1f} \star \sigma_{2f},\ (v_1, v_2)\,) \in \mathcal{V}_{k-j}[\![\cdot]\!] \subseteq \mathcal{V}_{k-2j}[\![\cdot]\!] \subseteq \mathcal{C}_{k-2j}[\![\cdot]\!]$ as needed.

⟨1⟩8. Case: Ty_Lambda.
Prove: $(\sigma, \gamma(\delta(\mathbf{fun}\, x : t \rightarrow e))) \in \mathcal{C}_k[\![\theta(t \multimap t')]\!]$.
Suffices: ??? By 6, to show $\ldots \in \mathcal{V}_k[\![\theta(t \multimap t')]\!]$.
Assume: Arbitrary $j < k$, $(\sigma_v, v) \in \mathcal{V}_j[\![\theta(t)]\!]$ such that $\sigma \star \sigma_v$ is defined.
Suffices: $(\sigma \star \sigma_v, \gamma(\delta(\mathbf{fun}\, x : t \rightarrow e))\, v) \in \mathcal{C}_j[\![\theta(t')]\!]$.
Suffices: $(\sigma \star \sigma_v, \gamma(\delta(e))[x/v]) \in \mathcal{C}_j[\![\theta(t')]\!]$.

⟨2⟩1. By induction, $[\![\Theta; \Delta; \Gamma, x : t \vdash e]\!]$.

⟨2⟩2. Instantiate it $\theta, j-1, \gamma[x \mapsto v], \sigma_v \star \sigma$.

⟨2⟩3. Hence, $(\sigma_v \star \sigma, \gamma[x \mapsto v](\delta(e))) \in \mathcal{C}_{j-1}[\![\theta(t)]\!]$.

⟨2⟩4. ??? By 3, we are done.

⟨1⟩9. Case: Ty_App.
Prove: $(\sigma, \gamma(\delta(\,e\, e'\,))) \in \mathcal{C}_k[\![\theta(t)]\!]$.
Assume: Arbitrary j and σ_r such that $\sigma \star \sigma_r$ defined.
Suffices: Show whole expression either reduces to **err** or a heap and expression in j steps.

⟨2⟩1. By induction,
1. $[\![\Theta; \Delta; \Gamma \vdash e : t' \multimap t]\!]$
2. $[\![\Theta; \Delta; \Gamma' \vdash e' : t']\!]$.

⟨2⟩2. Instantiate the first with $\theta, k, \delta, \gamma_e, \sigma_e$ as per usual definitions,
to conclude $(\sigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k[\![\theta(t' \multimap t)]\!]$.

⟨2⟩3. Instantiate *this* with j and $\sigma_{e'}$ to conclude $(\sigma = \sigma_e \star \sigma_{e'}, \gamma(\delta(\,e\, e'\,)))$ reduces to **err** or another heap and expression in j steps (using 3).

⟨2⟩4. Case: **err**
??? By Op_Context_Err, so too does the whole expression.
Since $j \leq k$ and σ_r (for 4.1) are arbitrary, $(\sigma, \gamma(\delta(\,e\, e'\,))) \in \mathcal{C}_k[\![\theta(t' \multimap t)]\!]$.

⟨2⟩5. CASE: j steps to another heap and expression.
By OP_CONTEXT, the whole expression does the same.
If it is not a value, we are done.
??? If it is $(\sigma_{ef}, \textbf{fun}\, x : t \to e_b) \in \mathcal{V}_{k-j}[\![\theta(t' \multimap t)]\!]$ by 4.1.

⟨2⟩6. SUFFICES: ??? By 4.1, to show $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(\,(\textbf{fun}\, x : t \to e_b)\, e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩7. Instantiate the second IH with $\theta, j, \delta, \gamma_{e'}, \sigma_{e'}$ defined as per usual.

⟨2⟩8. So, $(\sigma_{ef} \star \sigma_{e'}, \gamma_{e'}(\delta(e')))$ either reduces to **err** or a heap and expression in j steps.

⟨2⟩9. CASE: **err**
??? By OP_CONTEXT_ERR and 3, so too does the whole expression. Since $j \leq k$ and σ_r (for 4.1) are arbitrary, $(\sigma_{ef} \star \sigma_{e'}, \gamma(\delta(\,(\textbf{fun}\, x : t \to e_b)\, e'))) \in \mathcal{C}_{k-j}[\![\theta(t)]\!]$.

⟨2⟩10. CASE: j steps to another heap and expression.
By OP_CONTEXT and 3, the whole expression does the same.

⟨2⟩11. If it is not a value, we are done. ??? If it is, by definition of $(\sigma_{ef}, \textbf{fun}\, x : t \to e_b) \in \mathcal{V}_{k-j}[\![\theta(t' \multimap t)]\!]$, we have $(\sigma_{ef} \star \sigma_{e'f}, \gamma(\delta(\,(\textbf{fun}\, x : t \to e_b)\, v'))) \in \mathcal{C}_{k-2j}[\![\theta(t)]\!]$.

⟨1⟩10. CASE: TY_GEN.
PROVE: $(\sigma, \gamma(\delta(\textbf{fun}\, fc \to e))) \in \mathcal{C}_k[\![\theta(\forall\, fc.\, t)]\!]$.

⟨1⟩11. CASE: TY_SPC.
PROVE: $(\sigma, \gamma(\delta(e\,[f]))) \in \mathcal{C}_k[\![\theta(t\,[fc/f])]\!]$.

⟨1⟩12. CASE: TY_FIX.
PROVE: $(\sigma, \gamma(\delta(\textbf{fix}(g, x : t, e : t')))) \in \mathcal{C}_k[\![\theta(!(t \multimap t'))]\!]$.
SUFFICES: ??? to show $\ldots \in \mathcal{V}_k[\![!(\theta(t) \multimap \theta(t'))]\!]$, by 4.1.

⟨2⟩1. ASSUME: Arbitrary $j < k$ and $(\sigma, v) \in \mathcal{V}_j[\![\theta(t)]\!]$.

⟨2⟩2. SUFFICES: $(\sigma, \textbf{let Many}\; g = \textbf{fix}\,(g, x : t, e : t')\, \textbf{in}\, g\, v) \in \mathcal{C}_j[\![\theta(t')]\!]$.

⟨2⟩3. LET: $e_1 = e[g/\textbf{fun}\, x : t \to \textbf{let Many}\; g = \textbf{fix}\,(g, x : t, e : t')\, \textbf{in}\, g\, x]$.

⟨2⟩4. SUFFICES: ??? by 4.1, $(\sigma, (\textbf{fun}\, x : t \to e_1)\, v) \in \mathcal{C}_{j-1}[\![\theta(t')]\!]$.

⟨2⟩5. SUFFICES: ??? by 4.1, $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$.

⟨2⟩6. By induction, we have $[\![\Theta; \Delta, g : t \multimap t'; x : t \vdash e : t']\!]$.

⟨2⟩7. Instantiate this with $\theta, j-2, \delta[g \mapsto \textbf{fun}\, x : t \to e_1], \gamma = [x \mapsto v], \sigma$ (???).
PROVE: $(\sigma, \textbf{fun}\, x : t \to e_1) \in \mathcal{V}_{j-2}[\![\theta(t) \multimap \theta(t')]\!]$.

⟨3⟩1. SUFFICES: ??? by 4.1, $(\sigma', e_1[x/v']) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$ for arbitrary $(\sigma', v') \in \mathcal{V}_{j-2}[\![\theta(t)]\!]$.

⟨3⟩2. We can again use the induction hypothesis $[\![\Theta; \Delta, g : t \multimap t'; x : t \vdash e : t']\!]$.

⟨3⟩3. But since it's true for $\mathcal{C}_0[\![\cdot]\!]$ (base case), it's true by induction ???

⟨2⟩8. Lastly, we show $\delta(\gamma(e)) = e_1[x/v]$, which follows by their definitions, to conclude $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$.

⟨1⟩13. CASE: TY_VAR_LIN.
PROVE: $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

⟨2⟩1. $\Gamma = \{x : t\}$ by assumption of TY_VAR_LIN.

⟨2⟩2. SUFFICES: $(\sigma, \gamma(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$ by 3.

⟨2⟩3. By 2b, there exist $(\sigma_x, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$, such that $\sigma = \sigma_x$ and $\gamma = [x \mapsto v_x]$.

⟨2⟩4. ??? Hence, $(\sigma_x, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$, by 4.1.

⟨1⟩14. CASE: TY_VAR.
PROVE: $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$.

⟨2⟩1. $x : t \in \Delta$ and $\Gamma = \emptyset$ by assumption of TY_VAR.

⟨2⟩2. SUFFICES: $(\emptyset, \delta(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$ by 3 and 2b.

⟨2⟩3. By 2c, there exists v_x such that $(\emptyset, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$.

⟨2⟩4. ??? Hence, $(\emptyset, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$, by 4.1.

⟨1⟩15. CASE: TY_UNIT_INTRO.
PROVE: $(\sigma, \gamma(\delta(\,()\,))) \in \mathcal{C}_k[\![\theta(\mathbf{unit})]\!]$.

⟨1⟩16. CASE: TY_BOOL_TRUE, TY_BOOL_FALSE, TY_INT_INTRO, TY_ELT_INTRO.
Similar to TY_UNIT_INTRO.

5 Grammar Definition

m	$::=$			matrix expressions
	$\vert$	M		matrix variables
	$\vert$	$m + m'$		matrix addition
	$\vert$	$m\, m'$		matrix multiplication
	$\vert$	(m)	S	
f	$::=$			fractional capability
	$\vert$	fc		variable
	$\vert$	1		whole capability
	$\vert$	$\frac{1}{2} \cdot f$		
t	$::=$			linear type
	$\vert$	**unit**		unit
	$\vert$	**bool**		boolean (true/false)
	$\vert$	**int**		63-bit integers
	$\vert$	**elt**		array element
	$\vert$	f **arr**		arrays
	$\vert$	f **mat**		matrices
	$\vert$	$!t$		multiple-use type
	$\vert$	$\forall fc.t$	bind fc in t	frac. cap. generalisation
	$\vert$	$t \otimes t'$		pair
	$\vert$	$t \multimap t'$		linear function
	$\vert$	(t)	S	parentheses
p	$::=$			primitive
	$\vert$	**not**		boolean negation
	$\vert$	$(+)$		integer addition
	$\vert$	$(-)$		integer subtraction
	$\vert$	$(*)$		integer multiplication
	$\vert$	$(/)$		integer division
	$\vert$	$(=)$		integer equality
	$\vert$	$(\langle)$		integer less-than
	$\vert$	$(+.)$		element addition
	$\vert$	$(-.)$		element subtraction
	$\vert$	$(*.)$		element multiplication
	$\vert$	$(/.)$		element division
	$\vert$	$(=.)$		element equality
	$\vert$	$(<.)$		element less-than
	$\vert$	**set**		array index assignment
	$\vert$	**get**		array indexing
	$\vert$	**share**		share array
	$\vert$	**unshare**		unshare array
	$\vert$	**free**		free arrary
	$\vert$	**array**		Owl: make array
	$\vert$	**copy**		Owl: copy array
	$\vert$	**sin**		Owl: map sine over array

	|	**hypot**		Owl: $x_i := \sqrt{x_i^2 + y_i^2}$
	|	**asum**		BLAS: $\sum_i \|x_i\|$
	|	**axpy**		BLAS: $x := \alpha x + y$
	|	**dot**		BLAS: $x \cdot y$
	|	**rotmg**		BLAS: see its docs
	|	**scal**		BLAS: $x := \alpha x$
	|	**amax**		BLAS: $\mathrm{argmax}\, i : x_i$
	|	**setM**		matrix index assignment
	|	**getM**		matrix indexing
	|	**shareM**		share matrix
	|	**unshareM**		unshare matrix
	|	**freeM**		free matrix
	|	**matrix**		Owl: make matrix
	|	**copyM**		Owl: copy matrix
	|	**copyM_to**		Owl: copy matrix onto another
	|	**sizeM**		dimension of matrix
	|	**trnsp**		transpose matrix
	|	**gemm**		BLAS: $C := \alpha A^{T?} B^{T?} + \beta C$
	|	**symm**		BLAS: $C := \alpha AB + \beta C$
	|	**posv**		BLAS: Cholesky decomp. and solve
	|	**potrs**		BLAS: solve with given Cholesky
v	::=			values
	|	p		primitives
	|	x		variable
	|	$()$		unit introduction
	|	**true**		true
	|	**false**		false
	|	k		integer
	|	l		heap location
	|	el		array element
	|	**Many** v		!-introduction
	|	**fun** $fc \to v$		frac. cap. abstraction
	|	$v[f]$		frac. cap. specialisation
	|	(v, v')		pair introduction
	|	**fun** $x : t \to e$	bind x in e	abstraction
	|	**fix** $(g, x : t, e : t')$	bind $g \cup x$ in e	fixpoint
	|	(v)	S	parentheses
e	::=			expression
	|	p		primitives
	|	x		variable
	|	**let** $x = e$ **in** e'	bind x in e'	let binding
	|	$()$		unit introduction
	|	**let** $() = e$ **in** e'		unit elimination
	|	**true**		true

	$\mid$ $\mathbf{false}$		false
	$\mid$ $\mathbf{if}\ e\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2$		if
	$\mid$ k		integer
	$\mid$ l		heap location
	$\mid$ el		array element
	$\mid$ $\mathbf{Many}\ e$		!-introduction
	$\mid$ $\mathbf{let}\ \mathbf{Many}\ x = e\ \mathbf{in}\ e'$		!-elimination
	$\mid$ $\mathbf{fun}\ fc \to e$		frac. cap. abstraction
	$\mid$ $e[f]$		frac. cap. specialisation
	$\mid$ (e, e')		pair introduction
	$\mid$ $\mathbf{let}\ (a, b) = e\ \mathbf{in}\ e'$	bind $a \cup b$ in e'	pair elimination
	$\mid$ $\mathbf{fun}\ x : t \to e$	bind x in e	abstraction
	$\mid$ $e\ e'$		application
	$\mid$ $\mathbf{fix}\ (g, x : t, e : t')$	bind $g \cup x$ in e	fixpoint
	$\mid$ (e)	S	parentheses
C	$::=$		evaluation contexts
	$\mid$ $\mathbf{let}\ x = [-]\ \mathbf{in}\ e$	bind x in e	let binding
	$\mid$ $\mathbf{let}\ () = [-]\ \mathbf{in}\ e$		unit elimination
	$\mid$ $\mathbf{if}\ [-]\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2$		if
	$\mid$ $\mathbf{Many}\ [-]$		!-introduction
	$\mid$ $\mathbf{let}\ \mathbf{Many}\ x = [-]\ \mathbf{in}\ e$		!-elimination
	$\mid$ $\mathbf{fun}\ fc \to [-]$		frac. cap. abstraction
	$\mid$ $[-][f]$		frac. cap. specialisation
	$\mid$ $([-], e)$		pair introduction
	$\mid$ $(v, [-])$		pair introduction
	$\mid$ $\mathbf{let}\ (a, b) = [-]\ \mathbf{in}\ e$	bind $a \cup b$ in e	pair elimination
	$\mid$ $[-]e$		application
	$\mid$ $v[-]$		application
Θ	$::=$		fractional capability environment
	$\mid$ $\cdot$		
	$\mid$ Θ, fc		
Γ	$::=$		linear types environment
	$\mid$ $\cdot$		
	$\mid$ $\Gamma, x : t$		
	$\mid$ Γ, Γ'		
Δ	$::=$		intuitionistic types environment
	$\mid$ $\cdot$		
	$\mid$ $\Delta, x : t$		
σ	$::=$		heap
	$\mid$ $\{\}$		empty heap
	$\mid$ $\sigma \uplus \{l \mapsto_f m_{k_1,k_2}\}$		location l points to matrix m

StepsTo	::=		result of small step
	|	$\langle \sigma, e \rangle$	heap and expression
	|	**err**	error